12-2021

A Multigrid Homotopy Method For Computing Eigenfunctions of Differential Operators With Two-Dimensional Heterogeneity

Chelsea Drum

A MULTIGRID HOMOTOPY METHOD FOR COMPUTING EIGENFUNCTIONS OF DIFFERENTIAL OPERATORS WITH TWO-DIMENSIONAL HETEROGENEITY

by

Chelsea Drum

A Thesis
Submitted to the Graduate School,
the College of Arts and Sciences
and the School of Mathematics and Natural Sciences
of The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Master of Science

Approved by:

Dr. James Lambers, Committee Chair
Dr. Haiyan Tian
Dr. Huiqing Zhu

December 2021

ABSTRACT

In this thesis, we develop an accurate algorithm for computing the smallest eigenvalues of the self-adjoint spatial differential operator for the two-dimensional heat equation with a piecewise constant coefficient and periodic boundary conditions. The piecewise constant coefficient is created by the discontinuity of the diffusivity coefficient across the interfaces between two or more heterogeneous materials. Our method involves the combination of the well-known Inverse Iteration with a multigrid homotopy.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Dr. Lambers. I am grateful for his patience, encouragement, and guidance through each stage of the process. Many thanks to my committe members Dr. Tian and Dr. Zhu for their valuable time and feedback. I also wish to thank my family and friends for their unrelenting love and support.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure

LIST OF TABLES

Table

Chapter 1

INTRODUCTION

The fundamentally important partial differential equation (PDE) known as the heat equation is as follows:

$$u_t = \frac{1}{c(x,y)^2}\nabla \cdot (a(x,y)^2 \nabla u) \tag{1.1}$$

where

$$c(x,y)^2 = \text{specific heat coefficient}$$
$$a(x,y)^2 = \text{diffusion coefficient.}$$

We consider both the non-self-adjoint form

$$u_t = \alpha(x,y)^2 \nabla^2 u = \alpha(x,y)^2 (u_{xx} + u_{yy}) \tag{1.2}$$

and the self-adjoint form

$$u_t = \nabla \cdot (\alpha(x,y)^2 \nabla u) \tag{1.3}$$

on the domain $[0,2\pi] \times [0,2\pi]$ with

$$\alpha(x,y) = \begin{cases} \alpha_{11} & 0 \le x < \pi,\ 0 \le y < \pi \\ \alpha_{12} & \pi \le x < 2\pi,\ 0 \le y < \pi \\ \alpha_{21} & 0 \le x < \pi,\ \pi \le y < 2\pi \\ \alpha_{22} & \pi \le x < 2\pi,\ \pi \le y < 2\pi \end{cases} \tag{1.4}$$

Also known as a diffusion equation, (1.1) describes the diffusion of heat energy within a medium. Specifically, we are interested in the case of heat flux across two or more heterogeneous materials. Due to this heterogeneity, the coefficient α has discontinuities across the interfaces between materials. However, for each quadrant on the domain, the coefficient is constant. The non-self-adjoint form (1.2) is considered in the first two approaches, but the self-adjoint form (1.3) will be the focus of this thesis.

It is known that heat equations can be utilized for the modeling of many diffusive processes and phenomena. This specific heat equation with two-dimensional heterogeneity can apply to any such phenomena that have discontinuities at the interfaces. The discontinuities are created from the differences in diffusivity of the materials. This problem could model certain fluid mechanics, such as reservoir simulation where the coefficients represent rock

permeability [7]. This method could also be applied to electromagnetic wave propagation [5]. Under the umbrella of biological science, a few more applications include simulating blood flow patterns or even tumor growth [2].

In the 1-D case of this PDE, a highly accurate and efficient numerical method for computing its solution has been presented. This method utilized the Uncertainty Principle to estimate eigenvalues and then used the estimates to construct a basis of eigenfunctions for use with a spectral method [6]. In 2-D, previous work has been completed by Aurko for the case of only two homogenous materials [1]. He was able to develop an algorithm for computing the solution of the PDE by representing the solution as a linear combination of functions. Unlike the 1-D case, sinh and cosh were utilized in addition to sine and cosine.

In this thesis, we numerically approximate the eigenfunctions of the spatial differential operator of (1.3), a 2-D PDE with periodic boundary conditions. Our first approaches involved extending the previous research done in 1-D and 2-D for the non-self-adjoint spatial differential operator of (1.2). Due to their failure, our later approaches utilize iterative methods for a self-adjoint spatial differential operator of (1.3). Visualizations of the eigenfunctions of the operator play a key role in the development of our current method, Inverse Iteration with a Multigrid Homotopy.

This chapter has introduced a "bird's eye view" of our problem and how we will approach it. Chapter 2 will discuss the proposed problem in more mathematical detail. It will also address previous approaches that were attempted along with why they failed. Chapter 3 will present our numerical method for approximating the eigenfunctions of the spatial differential operator related to the PDE. Chapter 4 will include various numerical results obtained from the proposed method and finally, Chapter 5 will conclude the thesis and mention potential future work.

Chapter 2

BACKGROUND

2.1 The Discontinuous Eigenvalue Problem

As previously discussed, our 2-D PDEs with a discontinuous coefficient are (1.3), the self-adjoint, and (1.2), the non-self-adjoint, on the domain $[0,2\pi]\times[0,2\pi]$ with the piecewise constant coefficient (1.3). We first consider the non-self-adjoint PDE, which leads to the 2-D eigenvalue problem

$$-\alpha(x,y)^2(u_{xx}+u_{yy})=\lambda u, \qquad (x,y)\in[0,2\pi]^2.$$

For convenience, we define

$$L=-\alpha(x,y)^2\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)$$

to be the non-self-adjoint spatial differential operator. This operator was appealing due to the continuous first partial derivatives of its eigenfunctions. Using centered difference discretization, we have

$$L_N=-A_N(D_N^x+D_N^y)$$

with N points per dimension where A_N implements pointwise multiplication by α^2, D_N^x discretizes $\frac{\partial^2}{\partial x^2}$ using finite differences, and similar for D_N^y. After the first two approaches, we transitioned to the self-adjoint PDE

$$-\nabla\cdot(\alpha^2\nabla u)=\lambda u.$$

In this thesis, we will focus on the self-adjoint operator

$$-Lu=\frac{\partial}{\partial x}\left(\alpha^2\frac{\partial U}{\partial x}\right)+\frac{\partial}{\partial y}\left(\alpha^2\frac{\partial U}{\partial y}\right).$$

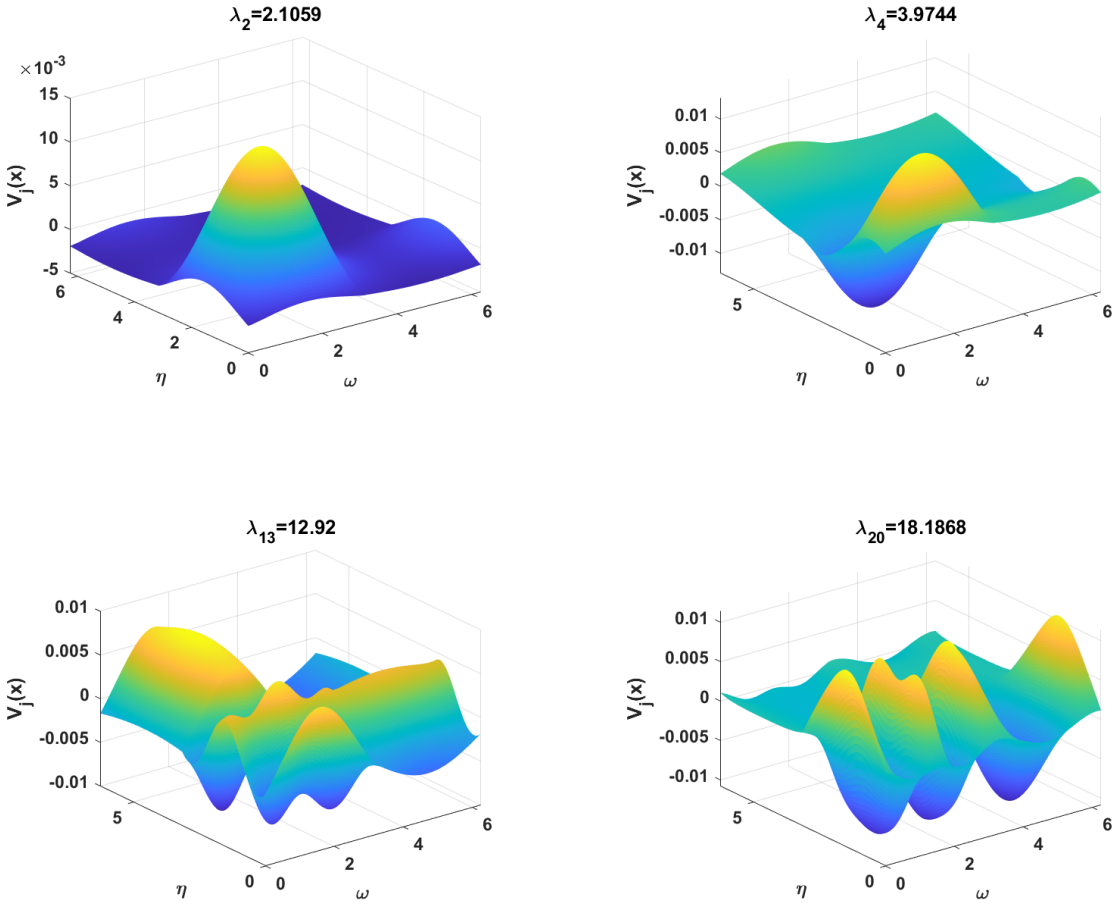
When discretized, we have

$$L_N={D_N^x}^T(A_N)D_N^x+{D_N^y}^T(A_N)D_N^y$$

with N points per dimension where A_N implements pointwise multiplication by α^2, D_N^x discretizes $\frac{\partial}{\partial x}$ using finite differences, and similar for D_N^y. The important distinction between the two operators is the fact that one operator is self-adjoint and the other is not.

Properties of the self-adjoint operator include continuity and periodicity of its individual eigenfunctions. This is represented in Figure 2.1 below. Another important property, that motivated our current approach, is a change in oscillation as the size of the eigenvalues is increased. This is also represented in Figure 2.1. The eigenvectors corresponding to the smaller eigenvalues are much less oscillatory than the larger ones. Of our interest, are the smaller, smoother eigenfunctions. This is due to the fact that when the solution of a PDE is expressed as an eigenfunction expansion, the dominant terms come from the smallest eigenvalues.

Figure 2.1: Various Eigenvectors of the Self-Adjoint Operator



2.2 Failed Approaches

2.2.1 Solution as a Series of Sines and Cosines

The first approach to computing the eigenfunctions of the non-self-adjoint spatial differential operator involves attempting to represent the solution of the eigenvalue problem as series of sines and cosines. This approach was motivatied by Sarah Wright's previous work in 1-D where the solution of the heat equation is presented as a truncated eigenfunction expansion, with each eigenfunction as a wave function that changes frequencies at the interfaces between materials.

For each quadrant $(i,j),\ \ i,j=1,2$ of the domain, let the eigenfunction $V_{ij}(x,y)$ satisfy the constant-coefficient PDE

$$-\alpha_{ij}^2(V_{xx}+V_{yy})=\lambda V.$$

The proposed form of the eigenfunction, after separation of variables, is

$$V_{ij}(x,y)=\Big(A_{ij}\cos(\omega_{ij}x)+B_{ij}\sin(\omega_{ij}x)\Big)\Big(C_{ij}\cos(\eta_{ij}y)+D_{ij}\sin(\eta_{ij}y)\Big)$$

where ω_{ij} and η_{ij} are the frequencies and $A_{ij},B_{ij},C_{ij},D_{ij}$ are the unknown amplitudes. Imposing the continuity and periodicity conditions on the eigenfunction results in 8 new equations that we rearrange, expand, and assert linear independence upon. An example of these equations is

$$V_{11}(0,y)=V_{12}(2\pi,y),\ \ 0\le y<\pi,$$

representing an enforced periodicity on the lower half of the domain. Rearranging and expanding for this condition, we obtain

$$\begin{aligned}0=\cos(\eta_{12}y)\big[&A_{12}C_{12}\cos(2\pi\omega_{12})+B_{12}C_{12}\sin(2\pi\omega_{12})\big]\\&+\sin(\eta_{12}y)\big[A_{12}D_{12}\cos(2\pi\omega_{12})+B_{12}D_{12}\sin(2\pi\omega_{12})\big]\\&-A_{11}C_{11}\cos(\eta_{11}y)-A_{11}D_{11}\sin(\eta_{11}y).\end{aligned}$$

Asserting linear independence, the coefficients of the functions of y, or x, are set equal to zero and 4 new equations are formed for each of the 8 equations. This condition yields

$$\begin{aligned}0&=A_{12}C_{12}\cos(2\pi\omega_{12})+B_{12}C_{12}\sin(2\pi\omega_{12})\\0&=A_{12}D_{12}\cos(2\pi\omega_{12})+B_{12}D_{12}\sin(2\pi\omega_{12})\\0&=A_{11}C_{11}=A_{11}D_{11}.\end{aligned}$$

This process is also executed for the partial derivatives of the eigenfunction. After obtaining all of the equations, they are used to form a matrix representing a homogeneous system for

the coefficients. Let **r** be a vector of linearly spaced values were the components r_i are the square root of eigenvalues when $A(r)$ is singular. This system of equations, matrix $A(r)$, is created for each value r_i. Then, for each $A(r)$, the smallest singular value is computed. The values of r for which the smallest singular value is zero indicate when the homogeneous system has a nontrivial solution. The collection of smallest singular values produced is then compared to the actual roots of the eigenalues.

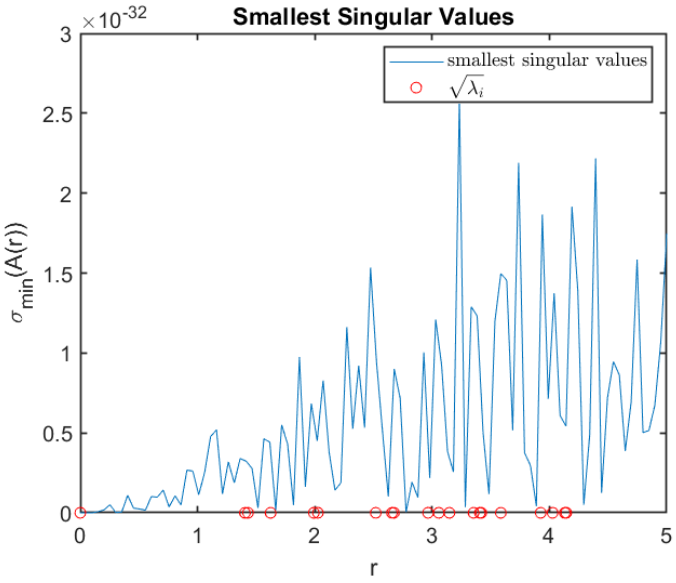


Figure 2.2: Smallest Singular Values of $A(r)$ vs. $\sqrt{\lambda_i}$

Our results are depicted here in Figure 2.2. Taking a closer look at the y-axis, it can be seen that only the trivial solution was found. It can be concluded that our system of equations was incorrect due to the inflexibility of the proposed form of the eigenfunction

$$V_{ij}(x,y) = \Big(A_{ij}\cos(\omega_{ij}x) + B_{ij}\sin(\omega_{ij}x)\Big)\Big(C_{ij}\cos(\eta_{ij}y) + D_{ij}\sin(\eta_{ij}y)\Big).$$

This motivates the next approach, representing the solution as a series of Bessel functions, as it incorporates more flexibility.

2.2.2 Solution as a Series of Fourier-Bessel Functions

Similar to the first, the second approach to computing the eigenfunctions of the non-self-adjoint spatial differential operator involves creating a homogeneous system of equations, computing smallest singular values, and comparing to the actual roots. The key difference is the ability to incorporate more flexibility in the eigenfunctions by representing the solution as a series of Fourier-Bessel functions rather than sines and cosines. This approach was

motivated by, and built on, previous work by Guidotti and Lambers [4]. As shown in their paper, the eigenvalue problem

$$-\triangle \varphi = \lambda \varphi \quad \text{in } \Omega$$

has the freespace eigenfunction

$$\varphi(x) = \int_{\mathbb{S}_r^{n-1}} e^{ix\xi} f(\xi)\, d\sigma_{\mathbb{S}_r^{n-1}}(\xi)$$

where $\mathbb{S}_r^{n-1}$ denotes a sphere of radius $r > 0$ in $\mathbb{R}^n$. $\varphi(x)$ satisfies the Laplacian, but not the boundary conditions. In order to satisfy the boundary conditions, $f(\xi)$ is replaced with its Fourier series

$$f(\theta) = \sum_{m\in\mathbb{Z}} \tau_m e^{im\theta}, \quad \theta \in [0, 2\pi].$$

From [4],we have that for any $r > 0$, one has that

$$\frac{1}{2\pi i^m} \int_0^{2\pi} e^{irx\xi_\theta} e^{im\theta}\, d\theta = J_m(r|x|)e^{im\theta_x}, \quad x \in \mathbb{R}^2, \quad \xi_\theta = (cos\theta, sin\theta).$$

The form of the eigenfunction then becomes

$$\varphi(x) = \sum_{m\in\mathbb{Z}} \tau_m J_m(r|x|)e^{im\theta_x}, \quad x \in \partial\Omega, \quad \text{with } \theta_x = arg(x_1 + ix_2).$$

We needed τ_m such that our periodic boundary conditions and continuity at the interfaces are satisfied. Therefore, the eigenfunction was set equal to an expansion of this form on each quadrant. For example, $\varphi_{11}(-\pi, y) = \varphi_{12}(\pi, y),\ -\pi \le y < 0$, asserts periodicity on the lower half of the domain.

Rearranging and expanding, we obtain

$$0 = \sum_{m=0}^{n} (\tau_m)_{11} J_m\left(r_{ij}\left|\sqrt{\pi^2 + y^2}\right|\right) e^{im\theta_{-\pi,y}} \tag{1a}$$

$$- \sum_{m=0}^{n} (\tau_m)_{12} J_m\left(r_{ij}\left|\sqrt{\pi^2 + y^2}\right|\right) e^{im\theta_{\pi,y}} \tag{1b}$$

Similarly, for periodicity on the upper half, we have

$$0 = \sum_{m=0}^{n} (\tau_m)_{21} J_m\left(r_{ij}\left|\sqrt{\pi^2 + y^2}\right|\right) e^{im\theta_{-\pi,y}} \tag{2a}$$

$$- \sum_{m=0}^{n} (\tau_m)_{22} J_m\left(r_{ij}\left|\sqrt{\pi^2 + y^2}\right|\right) e^{im\theta_{\pi,y}} \tag{2b}$$

We imposed these continuity and periodicity conditions on the eigenfunction, and its derivatives, resulting in 16 equations. However, we utilized 32 equations, where 16 are from

boundary points and 16 are from interior points. This idea to utilize interior points came from a paper by Betcke and Trefethen, where they represented the solution to the Laplace eigenvalue problem as a series of Fourier-Bessel equations. After finding the boundary points to be insufficient, due to ill-conditioning, they transitioned to utilizing both boundary and interior points during discretization. Then, the smallest singular values were only taken from the portion pertaining to boundary points [2]. We attempted to mimic this idea with our own problem by including both boundary and interior points during discretization, but then only finding the smallest singular values of the section pertaining to boundary points. Based on our 32 equations, we created a linear system of equations, matrix A, similar to the sines and cosines method.

The sparsity pattern of matrix A is shown in Figure 2.3. The blue areas represent nonzero values, while the white areas represent zero values. Examining the rows, there are 32 sections representing 32 equations. The first 16 were created with boundary points, and the latter with interior points. The height for each section of rows, or equation, was created by a chosen value of 20 grid points. Now examining the columns, they are split into 4 sections for the 4 quadrants of our domain. The width of each section of columns, or quadrant, represents the maximum Bessel order chosen.

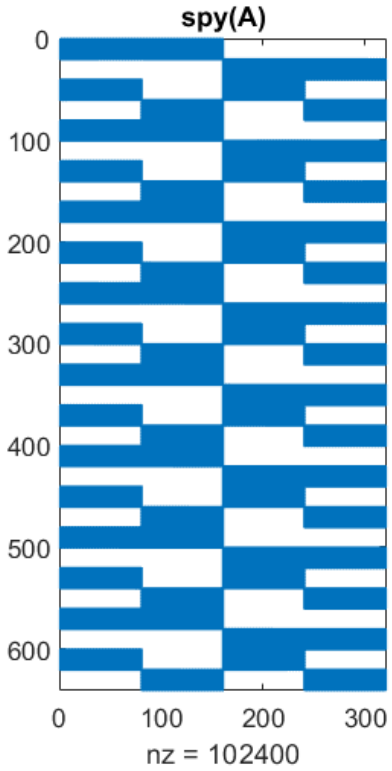


Figure 2.3: Sparsity Pattern of Matrix A

This matrix, $A(r_i)$, was created for each value r_i, where r is a vector of linearly spaced values. The smallest singular value of each matrix was computed and the collection of smallest singular values was then plotted against the actual roots of the eigenvalues. A psuedo-code for this process is shown below.

Choose a grid size N for the operator
L_N = `differentialoperator`(N)
V = matrix of L_N's 20 smallest eigenvectors produced by `eigs`
D = matrix produced by `eigs` where the diagonal is L_N's 20 smallest eigenvalues
rs = vector of equally spaced r values
Choose n for Bessel order
for j = 1 : length(r)
    for m = 0 : n
        Create matrix **A**, dependent on rs(j)
        Columns of A dependent on J_m
    end for
    $A = QR$
    s = smallest singular values of the portion of Q
        corresponding to boundary points
end for
`plot`(rs, s)

Given poor results, various modifactions to the method were attempted. For example, only boundary points were utilized before the incorporation of interior points. Another modification included centering the Fourier-Bessel functions at a location distant from the origin, rather than at the origin itself.

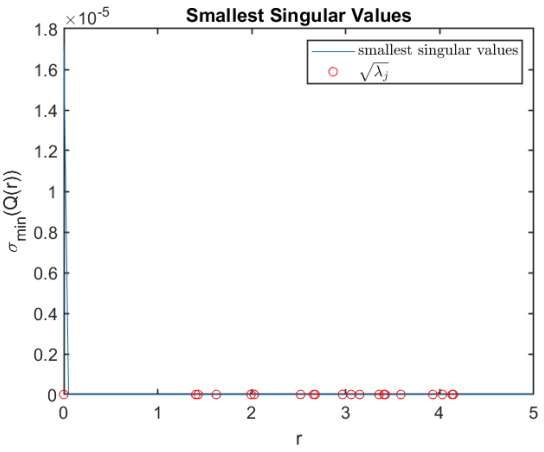


Figure 2.4: Smallest Singular Values of $A(r)$ vs. $\sqrt{\lambda_i}$

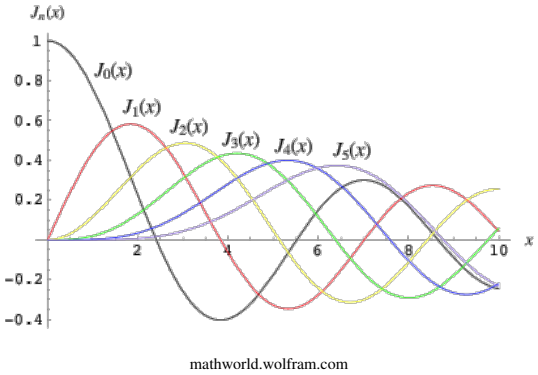


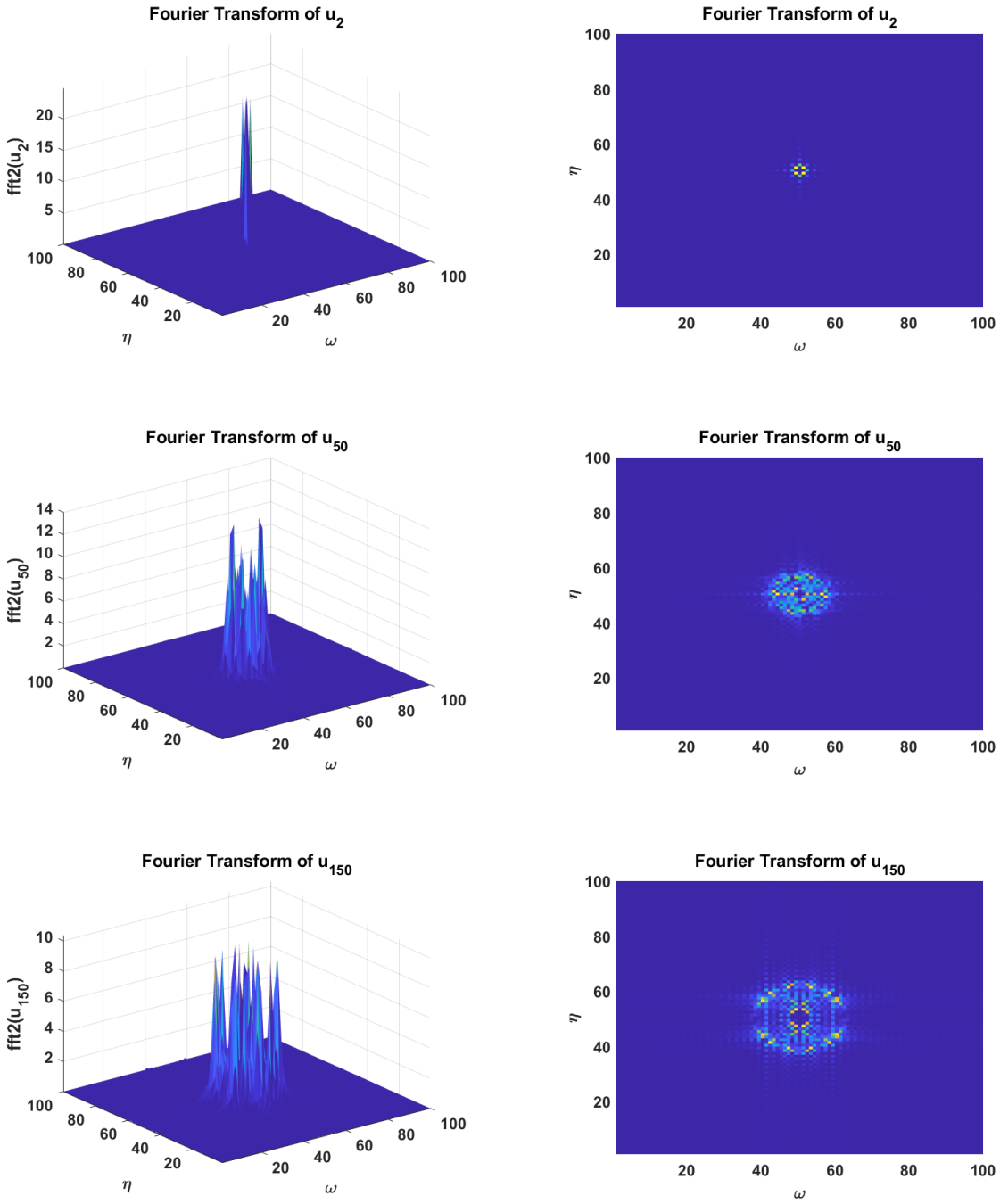
Figure 2.5: Bessel Function $J_n(x)$ for $n \in [1,5]$

The results are depicted in Figure 2.4. As shown, the smallest singular values are zero once again. This is due to an ill-conditioned matrix. We did not account for the Bessel function's proximity to zero in the region near the origin. This property of Bessel functions is shown in Figure 2.5.

2.2.3 Lanczos Iteration

In the third, and final, failed approach to computing the eigenfunctions, we transitioned to the self-adjoint spatial differential operator. Its symmetric property made it practical to utilize the Lanczos algorithm. However, before diving into this approach, we returned to examining and visualizing the properties of our eigenfunctions, to obtain the answers to certain questions. How do the eigenfunctions behave in terms of their Fourier transform? What, and where, are the dominant frequency components? To answer these, we reshaped the eigenvectors and visualized using `fft2` (2-D fast Fourier transform) in MATLAB. The results are displayed in Figure 2.6 below.

Figure 2.6: 2-D Fast Fourier Transform of Eigenvectors in Default `Surf` and Aerial View



The eigenvectors corresponding to the $2nd$, $50th$, and $150th$ eigenvalue are shown. For each eigenvector, there are two figures; the default `surf` view and an aerial view. From these figures, we can see the central location of the dominant frequency components. The central clusters found with the default surf view led to the examination of the aerial view. Due to the confirmed circular shape, we calculated the radius of the dominant frequency components for each eigenvector. We then plotted the radii in order to examine their behavior over the course of the first 200 eigenvectors.

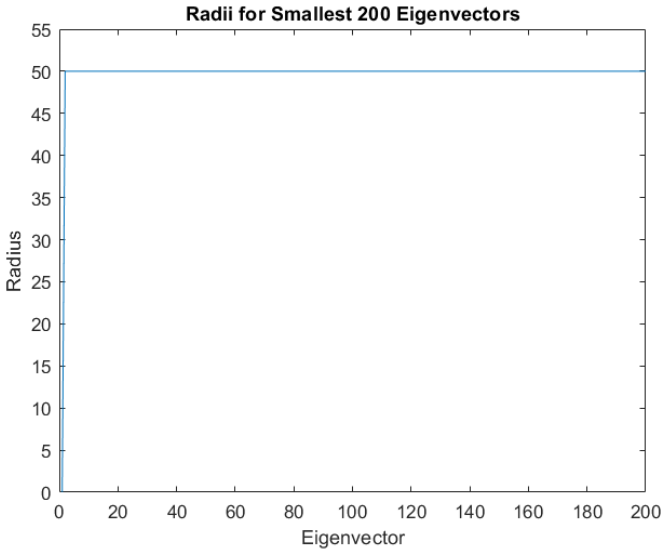


Figure 2.7: Radii of Dominant Frequency Components for the Smallest 200 Eigenvectors

From Figure 2.7, it can be seen that the radii of dominant components remains constant after the first few eigenvectors. We concluded the usage of a coarser grid would be justified. To take advantage of this, we utilized the radii as a suggestion for the number of grid points on the operator. The eigenfunctions corresponding to the smallest eigenvalues are fairly smooth and therefore working with a reduced grid should give a decent approximation for a larger grid. This multigrid idea is utilized in both the Lanczos method and the current methodology in the hope of reduced computational expense.

For the third approach, we utilized Lanczos iteration. As mentioned, the Lanczos algorithm was chosen to compute approximate eigenpairs of the self-adjoint operator due to its symmetric property. This method starts at a small operator size, but then doubles for each iteration. For the first operator size, the eigenpairs are computed using the Matlab `eig` function. The eigenvectors are then mapped to a finer grid through the padding of their Fourier transform with zeros. Moving to the next operator size, the previous eigenvectors are used as an intial guess for the Lanczos algorithm. The operator is shifted with the previous

operator's corresponsding eigenvalue. Lanczos is then applied to the shifted operator and an approximate eigenpair is found from the smallest Ritz value. This process is repeated for each operator size. A psuedo-code for this process can be found below.

p = maximum number of iterations where n determines operator size N
for n = 1 : p
    $N = 4 * 2^{n-1}$
    L_N = `differentialoperator`(N)
    if n == 1
        V = matrix of L_N's eigenvectors produced by `eig`
        D = matrix produced by `eig` where the diagonal is L_N's eigenvalues
    else
        k = number of eigenvalues desired
        for j = 1 : k
            Shift L_N with the j^{th} previous eigenvalue
            Initial guess: j^{th} column of previous eigenvectors
            Apply Lanczos to shifted L_N
            Output: approximate eigenpair from the smallest Ritz value
        end for
    end if
    Map the eigenvectors from a coarser grid to a finer grid
        (Pad their Fourier transforms with zeros)
end for

Table 2.1 below displays the results from the Lanczos Iteration method proposed. The method failed due to the numerical instability of the Lanczos algorithm. Only a few of the smallest eigenvalues were found.

Table 2.1: Lanczos algorithm results with $\alpha_{11} = 1$, $\alpha_{12} = 2$, $\alpha_{21} = 3$, $\alpha_{22} = 4$

Converged	*Eig*	Error
0.0000	0.0000	0.0000
2.0962	2.0958	0.0004
2.0962	3.3844	1.2883
3.9694	3.9467	0.0227
4.2613	4.2535	0.0078
6.4442	6.5961	0.1519
6.6145	8.2728	1.6584
8.3233	8.3274	0.0041
11.1180	9.3300	1.7880
13.6804	9.3598	4.3206
13.9312	10.6603	3.2709
14.1593	11.9238	2.2355
18.0789	13.2864	4.7925
21.4247	13.4464	7.9783
26.0252	13.6633	12.3618
42.3638	14.6004	27.7634

Chapter 3

METHODOLOGY

3.1 Visualizations

After the failure of the Lanczos method, we retreated back to visualizing our eigenfunctions in order to fully understand their behavior. Specifically, we examined the trajectory of the eigenvalues.

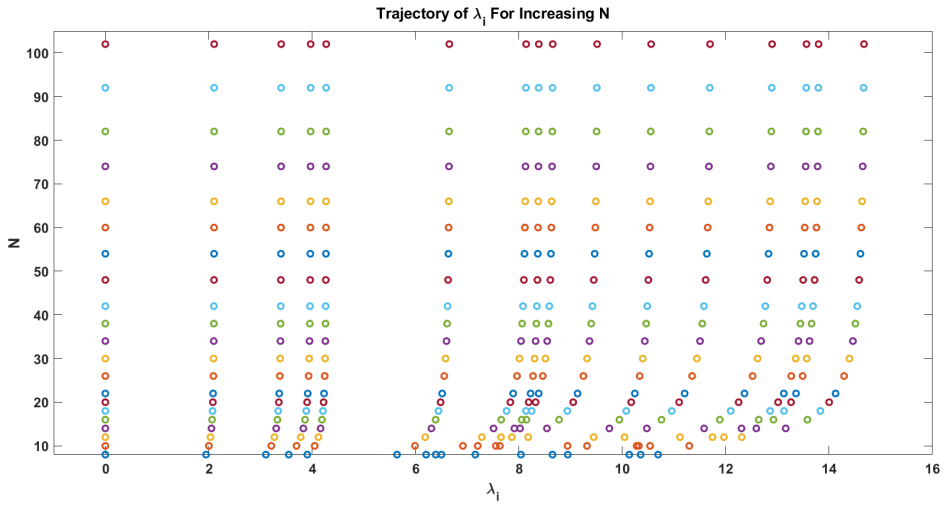


Figure 3.1: Trajectory of λ_i for $i \in [1,16]$ as N Increases

From Figure 3.1, we can see the trajectory of the smallest 16 eigenvalues as the operator size increases. The eigenvalues appear to converge for sufficient N.

Another idea that was introduced during this time is the usage of a homotopy. We desired an operator closer to a constant coefficient, since their eigenfunctions are known. Thus, the plan arose to utilize a homotopy in order to take intermediate steps from a constant coefficient to a piecewise constant coefficient. A new parameter, $t \in [0,1]$, was introduced with $t = 0$ representing a constant coefficient and $t = 1$ representing our piecewise constant coefficient. Specifically, for each coefficient α_{ij}, we used the equation

$$\alpha_{ij} = (1-t)h + t\alpha_{ij}$$

where t is the homotopy parameter and h is the harmonic average of the α_{ij} coefficients.

We examined the trajectory of the smallest 16 eigenvalues as the parameter t increases from 0 to 1, from a constant coefficient to our piecewise constant coefficient.

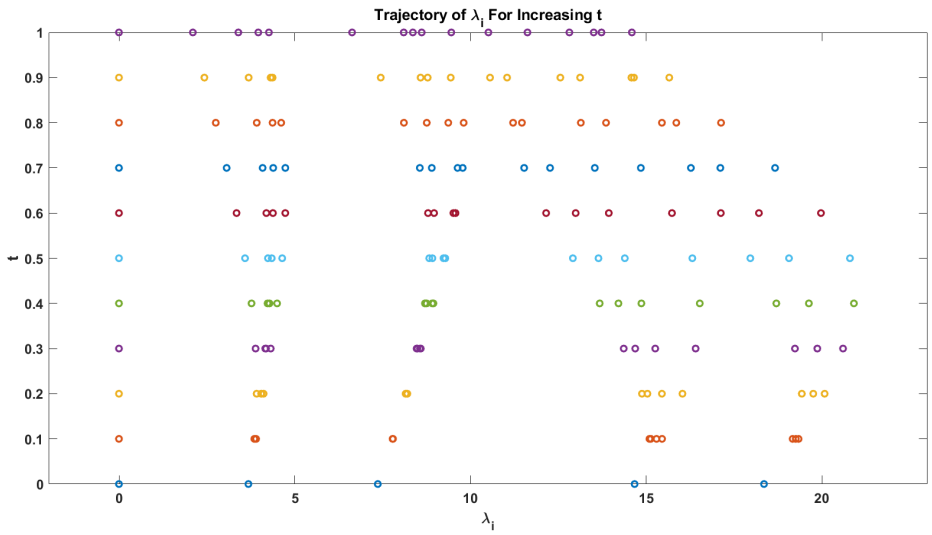


Figure 3.2: Trajectory of λ_i for $i \in [1,16]$ as t Increases

From Figure 3.2, we can see that each eigenvalue appears to have a unique path as it travels from the constant coefficient case to the piecewise constant coefficient case.

After seeing the trajectory of the eigenvalues as N increases and the trajectory of the eigenvalues as t increases, we were interested in finding the 'best path' for the eigenvalues while increasing N and t. We wanted to combine the multigrid idea introduced during our third approach with the new homotopy method of increasing t. We needed to find the best pattern of increasing N and t that would increase the accuracy and efficiency of any iterative method chosen. To do this, we needed to find a path that minimized the distance the eigenvalues had to travel from iteration to iteration. We created an interactive MATLAB program designed to allow users to choose between two choices: an increase in N or an increase in t. At first, the choices were chosen by the relative difference between the previous eigenvalues and the newly computed eigenvalues. However, if N was increased without t, the relative difference for the increase in t choice would increase with each jump in N. And similarly, if t was increased without N, the relative difference for the increase in N choice would increase for each jump in t. This meant we could not simply choose the step with the smallest relative difference because we needed both $t = 1$ and sufficient N for convergence of eigenvalues. This also meant we were unable to increase N or t too much without the other. Therefore, a third choice was created for the Matlab program: an increase in N and t at the same time.

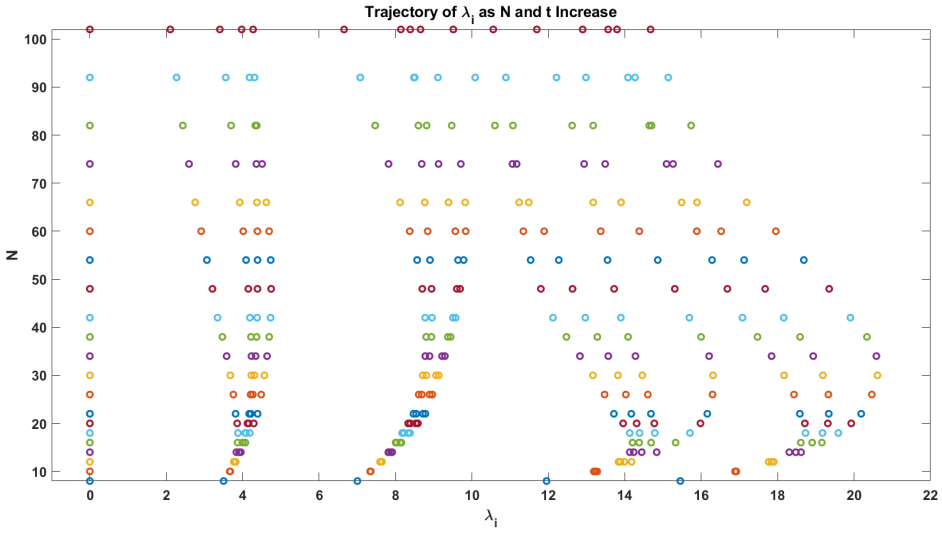


Figure 3.3: Trajectory of λ_i for $i \in [1,16]$ as N and t Increase Simultaneously

The best path found involves a small increase in N and t at the same time. Figure 3.3 displays the trajectory of the smallest 16 eigenvalues as N increases in 10% increments and t increases in .05 increments until $t = 1$, as those are our desired piecewise coefficients.

3.2 Inverse Iteration with a Multigrid Homotopy

Combining the multigrid idea with the homotopy parameter t, we found the best path for the smallest eigenvalues to take as they approach a sufficient N, or convergence of eigenvalues, with $t = 1$, our desired coefficients. The next step was to utilize an appropriate iterative method that would capitalize on this newly found path with both speed and accuracy. While Lanczos iteration is appealing due to the lower computational expense of matrix vector multiplication, Inverse iteration is also appealing due to its accuracy. We implemented both Lanczos and Inverse iteration in order to compare. Inverse iteration was found to be both faster and more accurate.

The multigrid approach allows for less computational expense as we travel from a small operator size to a larger operator size. As shown previously, in Figure 3.1, the eigenvalues converge for sufficient N, and therefore we need not travel to the exact operator size for precise approximations. The homotopy approach utilizes our knowledge of the constant coefficient case in order to provide a decent initial guess.

Now, to explain exactly how our method of Inverse Iteration with a Multigrid Homotopy works, we will start with a vector of equally spaced values of t and a vector of N values which increase by 10% for each index. For each loop, there is a 10% increase in N and

a .05 increase in t. The differential operator for specific N and t is created and Inverse Iteration is applied. The initial guess comes from the eigenvectors of the previous operator size and is orthogonalized against eigenvectors found for the current operator size. The differential operator is shifted with eigenvalues from the previous operator size. The output of Inverse Iteration is an approximate eigenpair for specific N and t. Before moving to the next operator size, the eigenvectors found are mapped from a coarser grid to a finer grid through the padding of their Fourier transforms with zeros. This allows the eigenvectors to be initial guesses for the next operator. The method is complete once $t = 1$, or in other words, we have reached our piecewise constant coefficient and sufficient size N. A psuedo-code for this process is shown below.

ts = vector of equally spaced t values ending at $t = 1$
Ns = vector of N values starting at 8 and increasing by 10% for each t value
for n = 1 :length(ts)
    N = Ns(n) ;
    t = ts(n) ;
    Apply homotopy to each alpha value
    L_N = `differentialoperator`(N)
    if n == 1
        V = matrix of L_N's eigenvectors produced by `eig`
        D = matrix produced by `eig` where the diagonal is L_N's eigenvalues
    else
        for j = 1:k
            Initial guess: j^{th} column of eigenvectors from the previous L_N
            Orthogonalize initial guess with a few eigenvectors found from the current L_N
            Shift L_N with the j^{th} eigenvalue from the previous L_N
            Apply Inverse Iteration
            Output: approximate eigenpair for specific N,t
        end for
    end if
    Map the eigenvectors from a coarser grid to a finer grid
    (Pad their Fourier transforms with zeros)
end for

Chapter 4

NUMERICAL RESULTS

The following numerical results were obtained through the method described in the previous chapter. The description of each table provides the α_{ij} coefficients chosen along with the starting t and N value. The Converged column refers to the smallest 16 eigenvalues found from our method, while the Eig column refers to the smallest 16 eigenvalues found from MATLAB's `eig` function. Also included is the error and the number of iterations used to find each eigenvalue. Tables 4.1-4.9 display results from a piecewise constant coefficent α_{ij}, while Tables 4.10-4.12 display results for the case of a smoothly varying coefficient. For all tables, the starting operator size is $N = 8$ and the Inverse iteration tolerance is set to $1e{-}10$. This tolerance is between the previous iteration's eigenvector and the current iteration's eigenvector. It can be seen that with an increased initial t value, there is a significant decrease in the number of iterations required per eigenvalue. With the increase in t, the accuracy tends to stay the same for each case of coefficients. The accuracy and expense for each piecewise constant case of coefficients is analagous in comparison to one another. The results also display a disparity in accuracy between the piecewise constant coefficient cases and the smoothly varying coefficient cases.

Table 4.1: Initial homotopy value $t = 0$ with $\alpha_{11} = 1$, $\alpha_{12} = 2$, $\alpha_{21} = 3$, $\alpha_{22} = 4$

Converged	*Eig*	error	k
0.0000	0.0000	0.0000	32
2.1051	2.1051	0.0000	40
3.4015	3.4014	0.0000	40
4.0833	3.9720	0.1112	40
4.2706	4.2719	0.0013	40
6.6533	6.6532	0.0001	40
8.1401	8.1390	0.0011	40
8.3845	8.3844	0.0000	40
8.6539	8.6536	0.0003	40
9.5130	9.5124	0.0006	40
10.5580	10.5575	0.0006	40
11.6999	11.6975	0.0024	40
12.8973	12.8973	0.0000	40
13.6435	13.5628	0.0808	40
13.7568	13.7968	0.0400	40
14.6748	14.6744	0.0004	40

Table 4.2: Initial homotopy value $t = .25$ with $\alpha_{11} = 1$, $\alpha_{12} = 2$, $\alpha_{21} = 3$, $\alpha_{22} = 4$

Converged	*Eig*	error	k
0.0000	0.0000	0.0000	19
2.1032	2.1032	0.0000	30
3.3980	3.3979	0.0000	30
3.9750	3.9668	0.0082	30
4.2680	4.2681	0.0001	30
6.6404	6.6404	0.0001	30
8.1167	8.1158	0.0009	30
8.3701	8.3700	0.0001	30
8.6287	8.6284	0.0003	30
9.4783	9.4777	0.0006	30
10.5291	10.5285	0.0006	30
11.6543	11.6515	0.0029	30
12.8466	12.8465	0.0001	30
13.5338	13.5285	0.0053	30
13.7534	13.7548	0.0014	30
14.6245	14.6242	0.0004	30

Table 4.3: Initial homotopy value $t = .5$ with $\alpha_{11} = 1$, $\alpha_{12} = 2$, $\alpha_{21} = 3$, $\alpha_{22} = 4$

Converged	*Eig*	error	k
0.0000	0.0000	0.0000	20
2.0970	2.0970	0.0000	20
3.3866	3.3866	0.0000	20
4.0097	3.9499	0.0598	20
4.2554	4.2559	0.0005	20
6.5991	6.5989	0.0001	20
8.0420	8.0409	0.0011	20
8.3234	8.3231	0.0004	20
8.5474	8.5471	0.0003	20
9.3658	9.3653	0.0005	20
10.4353	10.4348	0.0005	20
11.5073	11.5035	0.0038	20
12.6851	12.6849	0.0002	20
13.4212	13.4082	0.0130	20
13.6191	13.6223	0.0033	20
14.4618	14.4615	0.0003	20

Table 4.4: Initial homotopy value $t = 0$ with $\alpha_{11} = 2$, $\alpha_{12} = 2$, $\alpha_{21} = 4$, $\alpha_{22} = 4$

Converged	*Eig*	error (1e-5)	k
0.0000	0.0000	0.0000	33
5.9150	5.9150	0.0050	40
6.5954	6.5954	0.0675	40
6.5954	6.5954	0.0675	40
8.5774	8.5774	0.0012	40
14.8763	14.8763	0.2676	40
14.8763	14.8763	0.2676	40
17.2493	17.2492	0.0581	40
17.2493	17.2492	0.0581	40
19.3099	19.3099	0.1012	40
19.3099	19.3099	0.1012	40
25.6685	25.6685	0.0363	40
29.1212	29.1212	0.5291	40
29.1212	29.1212	0.5291	40
30.9424	30.9424	0.4268	40
35.8093	35.8093	0.0044	40

Table 4.5: Initial homotopy value $t = .25$ with $\alpha_{11} = 2$, $\alpha_{12} = 2$, $\alpha_{21} = 4$, $\alpha_{22} = 4$

Converged	*Eig*	error	k
0.0000	0.0000	0.0000	27
5.9093	5.9093	0.0000	30
6.5913	6.5913	0.0000	30
6.5913	6.5913	0.0000	30
8.5729	8.5729	0.0000	30
14.8553	14.8553	0.0000	30
14.8553	14.8553	0.0000	30
17.2371	17.2371	0.0000	30
17.2371	17.2371	0.0000	30
19.2680	19.2680	0.0000	30
19.2680	19.2680	0.0000	30
25.5829	25.5829	0.0000	30
29.0528	29.0528	0.0000	30
30.8430	29.0528	1.7902	30
35.7061	30.8430	4.8631	30
35.7061	35.7061	0.0000	30

Table 4.6: Initial homotopy value $t = .5$ with $\alpha_{11} = 2$, $\alpha_{12} = 2$, $\alpha_{21} = 4$, $\alpha_{22} = 4$

Converged	*Eig*	error	k
0.0000	0.0000	0.0000	20
5.8909	5.8909	0.0000	20
6.5779	6.5779	0.0000	20
6.5779	6.5779	0.0000	20
8.5584	8.5584	0.0000	20
14.7875	14.7874	0.0000	20
14.7875	14.7874	0.0000	20
17.1974	17.1974	0.0000	20
17.1974	17.1974	0.0000	20
19.1333	19.1333	0.0000	20
19.1333	19.1333	0.0000	20
25.3072	25.3072	0.0000	20
28.8330	28.8330	0.0000	20
30.5208	28.8330	1.6879	20
35.3748	30.5208	4.8540	20
35.3748	35.3748	0.0000	20

Table 4.7: Initial homotopy value $t = 0$ with $\alpha_{11} = 1$, $\alpha_{12} = 3$, $\alpha_{21} = 5$, $\alpha_{22} = 1$

Converged	*Eig*	error	k
0.0000	0.0000	0.0000	32
1.8758	1.8758	0.0000	40
2.8528	2.8238	0.0290	40
2.8546	2.8762	0.0215	40
3.9622	3.9562	0.0059	40
3.9741	3.9784	0.0043	40
4.6731	4.6730	0.0000	40
4.6754	4.6754	0.0000	40
6.9730	6.9730	0.0000	40
7.6883	7.6883	0.0000	40
9.3903	9.0200	0.3703	40
9.4538	9.4532	0.0007	40
9.6297	9.6198	0.0099	40
10.9798	9.6317	1.3481	40
12.2315	10.5664	1.6652	40
12.2602	10.9991	1.2611	40

Table 4.8: Initial homotopy value $t = .25$ with $\alpha_{11} = 1$, $\alpha_{12} = 3$, $\alpha_{21} = 5$, $\alpha_{22} = 1$

Converged	*Eig*	error	k
0.0000	0.0000	0.0000	27
1.8746	1.8746	0.0000	30
2.8388	2.8244	0.0144	30
2.8631	2.8727	0.0096	30
3.9482	3.9482	0.0000	30
3.9798	3.9798	0.0000	30
4.6643	4.6643	0.0000	30
4.6682	4.6682	0.0000	30
6.9631	6.9631	0.0000	30
7.6699	7.6699	0.0000	30
9.1221	8.9859	0.1362	30
9.3907	9.3907	0.0000	30
9.6439	9.5774	0.0665	30
10.9325	9.6445	1.2879	30
12.1779	10.5103	1.6676	30
12.2208	10.9327	1.2881	30

Table 4.9: Initial homotopy value $t = .5$ with $\alpha_{11} = 1$, $\alpha_{12} = 3$, $\alpha_{21} = 5$, $\alpha_{22} = 1$

Converged	*Eig*	error	k
0.0000	0.0000	0.0000	20
1.8708	1.8708	0.0000	20
2.8210	2.8208	0.0002	20
2.8607	2.8609	0.0001	20
3.9304	3.9304	0.0000	20
3.9759	3.9759	0.0000	20
4.6367	4.6367	0.0000	20
4.6439	4.6439	0.0000	20
6.9299	6.9299	0.0000	20
7.6108	7.6108	0.0000	20
8.9721	8.8765	0.0956	20
9.2528	9.2528	0.0000	20
9.5716	9.4375	0.1340	20
9.6238	9.6249	0.0011	20
10.4935	10.3302	0.1633	20
12.0133	10.7211	1.2922	20

Table 4.10: Initial homotopy value $t = 0$ with smoothly varying coefficients defined by $f = 1 + (1/2)sin(X)cos(Y)$

Converged	*Eig*	error (1e-8)	k
0.0000	0.0000	0.0000	31
0.9617	0.9617	0.0003	40
0.9662	0.9662	0.0004	40
0.9662	0.9662	0.0000	40
0.9706	0.9706	0.0000	40
1.8234	1.8234	0.0000	40
1.8870	1.8870	0.0001	40
1.8871	1.8871	0.0000	40
1.9550	1.9550	0.0000	40
3.7939	3.7939	0.0297	40
3.8557	3.8557	0.1256	40
3.8557	3.8557	0.0970	40
3.9129	3.9129	0.0000	40
4.1147	4.1147	0.0028	40
4.2121	4.2121	0.0081	40
4.2122	4.2122	0.0088	40

Table 4.11: Initial homotopy value $t = .25$ with smoothly varying coefficients defined by $f = 1 + (1/2)sin(X)cos(Y)$

Converged	*Eig*	error (1e-7)	k
0.0000	0.0000	0.0000	26
0.9611	0.9611	0.0009	30
0.9656	0.9656	0.0009	30
0.9656	0.9656	0.0000	30
0.9701	0.9701	0.0000	30
1.8222	1.8222	0.0000	30
1.8858	1.8858	0.0000	30
1.8859	1.8859	0.0000	30
1.9538	1.9538	0.0000	30
3.7846	3.7846	0.0825	30
3.8465	3.8465	0.1162	30
3.8465	3.8465	0.0315	30
3.9037	3.9037	0.0003	30
4.1055	4.1055	0.0075	30
4.2028	4.2028	0.0072	30
4.2031	4.2031	0.0050	30

Table 4.12: Initial homotopy value $t = .5$ with smoothly varying coefficients defined by $f = 1 + (1/2)sin(X)cos(Y)$

Converged	*Eig*	error (1e-6)	k
0.0000	0.0000	0.0000	20
0.9592	0.9592	0.0074	20
0.9637	0.9637	0.0076	20
0.9637	0.9637	0.0003	20
0.9682	0.9682	0.0003	20
1.8180	1.8180	0.0000	20
1.8816	1.8816	0.0000	20
1.8820	1.8820	0.0000	20
1.9501	1.9501	0.0000	20
3.7545	3.7544	0.7050	20
3.8166	3.8166	0.7811	20
3.8167	3.8167	0.0544	20
3.8741	3.8741	0.0576	20
4.0756	4.0756	0.0587	20
4.1727	4.1727	0.0125	20
4.1737	4.1737	0.0119	20

Chapter 5

CONCLUSION

The bulk of this research was filled with many trials and errors. It was known how to solve the one-dimensional version of our PDE, but transferring to two-dimensions brought its own difficulties. We sought an accurate and efficient algorithm for computing the eigenfunctions of the differential operator produced by our 2-D heat equation with a piecewise constant coefficient and periodic boundary conditions. With the first, non-self-adjoint operator, we attempted to transfer our 1-D knowledge by representing the solution as a series of sines and cosines. Due to its inflexibility, we transitioned to representing our solution as a series of Fourier-Bessel functions. However, only the trivial solution was found once again, due to the Bessel functions' proximity to zero near the origin.

Moving forward, we switched to the self-adjoint operator in order to utilize its symmetric property. We applied Lanczos iteration with the hope of decent approximate eigenfunctions; however, the numerical instability of Lanczos prevailed. We returned to visualizing the eigenfunctions in the hopes of learning more about their behavior. We found that, with an increasing operator size N, the trajectories of the smaller eigenvalues converge given sufficient N. We also found that the smaller eigenvalues appear to have a unique path from an operator of the constant coefficient case to an operator of the piecewise constant coefficient case. We utilized this knowledge to introduce a multigrid homotopy method that could be applied with an iterative method. Due to its tendency of acccuracy, Inverse Iteration was chosen to be paired with the multigrid homotopy method.

This approach is able to accurately find the smallest eigenvalues of our differential operator; however, it is much more expensive than in the 1-D method. Also, this approach does not take into account the true nature of our differential operator. Our method can be applied to both the piecewise constant coefficient case and a smoothly varying coefficient case. Future work should attempt to utilize our knowledge of the eigenvalues and consequently improve our efficiency. Other than improving the computation expense and speed, one other prospect we have for future work includes extrapolation. In the Inverse Iteration framework, we would take the eigenvalues found so far, fit them with an interpolant, and then utilize that for a better initial guess.

Appendix A

MATLAB Code

A.1 Inverse Iteration With a Multigrid Homotopy

InverseItIncreaseNandt.m

```
ts=.5:.05:1;
Ns=[8];
for i=1:length(ts)
    N=ceil(1.1*Ns(i));
    eo=(-1)^N;
    if eo==-1
        N=N+1;
    end
    Ns=[Ns N];
end
ks=zeros(16,1);
for n=1:length(ts)
    N=Ns(n);
    alpha11=1;  alpha12=3;  alpha21=5;  alpha22=1;
    havg=(4/((1/alpha11)+(1/alpha12)+(1/alpha21)+(1/alpha22)));
    t=ts(n);
    %t=t^1/2;
    a11=(1-t)*havg+t*alpha11;
    a12=(1-t)*havg+t*alpha12;
    a21=(1-t)*havg+t*alpha21;
    a22=(1-t)*havg+t*alpha22;
    L_N=diffoperator(N,a11,a12,a21,a22);
    %L_N=diffoperatorSVC(N);
    ogLN=L_N;
    if n==1
        L_N=full(L_N);
        [V,D]=eig(L_N);
        es=diag(D);
        es=es(1:16);
        evs=V(:,1:16);
    else
        es=[ ];
        ef1=ones(N*N,1)/N;
        evs=[ ];
        for j=1:16
            %initial guess
            rk=pevs(:,j);
            %Orthogonalize initial guess w/ only a few previous eigvecs
            if j~=1
                fevs4or=[ ];
                for b=1:j-1
                    Rq=(evs(:,b)'*ogLN*evs(:,b))/(evs(:,b)'*evs(:,b));
                    dis=abs((Rq-peigs(b))/peigs(b));
                    if dis < .25
```

```
                    fevs4or=[fevs4or evs(:,b)];
                end
            end
            if isempty(fevs4or)==0
                Pk=fevs4or*inv(fevs4or'*fevs4or)*fevs4or';
                Pkp=eye(size(Pk))-Pk;
                rk=Pkp*rk;
            end
        end
        %shifted L_N
        L_N=ogLN-peigs(j)*speye(size(ogLN));
        %INVERSE ITERATION
        evp=1;
        evn=0;
        k=0;
        while norm(evp-evn) > 1e-10
            k=k+1;
            evp=evn;
            ev=L_N\rk;
            e=(ev'*L_N*ev)/(ev'*ev);
            evn=ev/norm(ev);
        end
        ks(j)=ks(j)+k;
        %eigenvalues w/shift
        es=[es; e+peigs(j)];
        %eigenvectors
        evs=[evs ev];
    end
    ks;
end
pevs=[ ];
if n~=length(ts)+1
    for c=1:16
        ev=reshape(evs(:,c),[N,N]);
        ev=fft2(ev);
        ev2=zeros(Ns(n+1));
        s=Ns(n+1)-(N-(N/2+2));
        ev2(1:N/2+1,1:N/2+1)=ev(1:N/2+1,1:N/2+1);
        ev2(1:N/2+1,s:Ns(n+1))=ev(1:N/2+1,N/2+2:N);
        ev2(s:Ns(n+1),1:N/2+1)=ev(N/2+2:N,1:N/2+1);
        ev2(s:Ns(n+1),s:Ns(n+1))=ev(N/2+2:N,N/2+2:N);
        ev2=ev2./4;
        ev2=real(ifft2(ev2));
        ev2=reshape(ev2,[ ],1);
        pevs=[pevs ev2];
    end
end
peigs=es;
end
[V,D]=eig(full(ogLN));
d=diag(D);
d=d(1:16);
for y=1:16
    if isnan(es(y))==1
        es(y)=0.0000;
    end
end
[sort(abs(es)) d];
```

A.2 Self-Adjoint Differential Operator

differentialoperator.m

```
function L_N=diffoperator(N,a11,a12,a21,a22)
%function that creates a matrix of a differential operator using a
%specified number of gridpoints and input values

%create NxN matrix that has 2x2 block structure where each block is N/2 x N/2
%with each entry in the i,j block is set equal to alpha_ij^2
%A_N
A=(a11^2)*ones(N/2);
B=(a12^2)*ones(N/2);
C=(a21^2)*ones(N/2);
D=(a22^2)*ones(N/2);
Atilde=[ A  B ; C  D ];
Atilde=reshape(Atilde,N^2,1);
A_N=spdiags(Atilde,0,N^2,N^2);
%D_N
deltax=2*pi/N;
%create NxN matrix d with -1s on the main diagonal, 1s on the upper
%bidiagonal, and a 1 in the bottom left corner
d=zeros(N);
for i=1:N
    for j=1:N
        if i==j
            d(i,j)=-1;
        end
        if i-j==-1
            d(i,j)=1;
        end
    end
end
d(N,1)=1;
d=sparse(d);
%multiply matrix d by 1/(2pi/N)^2
D=(1/deltax)*d;
%create NxN identity matrix
I=speye(N);
%D_N^y
%Kronecker product of D and I
D_N_x=kron(D,I);
%D_N^y
%Kronecker product of I and D
D_N_y=kron(I,D);
%now we can compute L_N
L_N=transpose(D_N_x)*A_N*D_N_x+transpose(D_N_y)*A_N*D_N_y;
end
```

A.3 Self-Adjoint Differential Operator With Smoothly Varying Coefficients

diffoperatorSVC.m

```
function L_N=differentialoperatorSVC(N)
%function that creates a matrix of a differential operator using a
%specified number of gridpoints
xv=transpose(linspace(0,2*pi,N+1));
xv=xv(1:N);
yv=xv;
[X,Y]=meshgrid(xv,yv);
X=reshape(X,[ ],1);
Y=reshape(Y,[ ],1);
%function f representing the smoothly varying coefficients
f=1+(1/2)*sin(X).*cos(Y);
Atilde=reshape(f,N^2,1);
A_N=spdiags(Atilde,0,N^2,N^2);
%D_N
deltax=2*pi/N;
%create NxN matrix d with -1s on the main diagonal, 1s on the upper
%bidiagonal, and a 1 in the bottom left corner
d=zeros(N);
for i=1:N
    for j=1:N
        if i==j
            d(i,j)=-1;
        end
        if i-j==-1
            d(i,j)=1;
        end
    end
end
d(N,1)=1;
d=sparse(d);
%multiply matrix d by 1/(2pi/N)^2
D=(1/deltax)*d;
%create NxN identity matrix
I=speye(N);
%D_N^y
%Kronecker product of D and I
D_N_x=kron(D,I);
%D_N^y
%Kronecker product of I and D
D_N_y=kron(I,D);
%now we can compute L_N
L_N=transpose(D_N_x)*A_N*D_N_x+transpose(D_N_y)*A_N*D_N_y;
end
```

BIBLIOGRAPHY

[1] Aurko, Abdullah Muheel Momit, "Eignefunctions for Partial Differential Equations on Two-Dimensional Domains With Piecewise Constant Coefficients" (2017). Master's Theses. 316. https://aquila.usm.edu/masters_theses/316 .

[2] Betcke, Timo, and Lloyd N. Trefethen. "Reviving the Method of Particular Solutions." SIAM Review 47, no. 3 (2005): 469–91. https://doi.org/10.1137/s0036144503437336.

[3] Garon, Elyse M., and James V. Lambers. "Modeling the Diffusion of Heat Energy within Composites of Homogeneous Materials Using the Uncertainty Principle." Computational and Applied Mathematics 37, no. 3 (2017): 2566–87. https://doi.org/10.1007/s40314-017-0465-6.

[4] Guidotti, Patrick, and James V. Lambers. "Eigenvalue Characterization and Computation for the Laplacian on General 2-D Domains." Numerical Functional Analysis and Optimization 29, no. 5-6 (2008): 507–31. https://doi.org/10.1080/01630560802099233.

[5] Lambers, James V. "A Spectral Time-Domain Method for Computational Electrodynamics." Numerical Mathematics and Advanced Applications 2009, 2010, 561–69. https://doi.org/10.1007/978-3-642-11795-4_60.

[6] Long, S. D., Sheikholeslami, S., Lambers, J. V., Walker, C. "Diagonalization of 1-D Differential Operators With Piecewise Constant Coefficients Using the Uncertainty Principle" (2019). Mathematics and Computers in Simulation, 156, 194-226. https://aquila.usm.edu/fac_pubs/15542.

[7] Peaceman, Donald W. Fundamentals of Numerical Reservoir Simulation. Amsterdam: Elsevier, 1977.

[8] Wise, S.M., J.S. Lowengrub, H.B. Frieboes, and V. Cristini. "Three-Dimensional Multispecies Nonlinear Tumor Growth—I." Journal of Theoretical Biology 253, no. 3 (2008): 524–43. https://doi.org/10.1016/j.jtbi.2008.03.027.